The SpacePy Team:
Steve Morley, Josef Koller, Dan Welling, Brian Larsen and Mike Henderson
(ISR-1, Space Science and Application, LANL)
email: spacepy-info@lanl.gov

SpacePy

Data analysis and visualization tools for the space sciences

What is SpacePy?

SpacePy is a package of data analysis and visualization tools. It includes implementations of widely used empirical models (plasmapause, magnetopause), statistical techniques used frequently in space science (e.g. superposed epoch analysis), and interfaces to advanced tools such as electron drift shell calculations for radiation belt studies. SpacePy also provides analysis and visualization tools for components of the Space Weather Modeling Framework (e.g. BATS-R-US, RAM-SCB).

SpacePy lets you work easily with different time formats, coordinate systems and file types. But it's not just for data – model results are easily parsed and handled in SpacePy. In fact, if you use the Space Weather Modeling Framework, most of the components are supported in SpacePy.

So far, the SpacePy team have been using the package to do a lot of science: radiation belt modeling; statistical studies of the radiation belts using GPS; L* calculations; analysis and visualization of BATS-R-US runs (coupled with other codes); even just rapid display of OMNI data in meetings!

So what's in SpacePy, and what does the name mean?

SpacePy is written in Python (hence the name) and, as such, has as its foundation the world's 7th most popular programming language! For comparison, MatLab is #18 and IDL isn't in the top 50.

This level of use brings a wide community all working on developing for the language. Python has great support for numerics, statistics, plotting, regular expressions … just about anything you need. Python is a scripting language and has great bindings to almost every other language. Including C, C++, Fortran, R, etc. is easy.

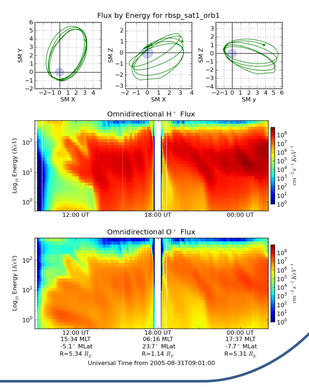
Since Python is a good "glue" language, it's well placed for use on big projects, but it's easy and fast for small jobs too. SpacePy was developed so that the routine analyses we do in space science are as easy as possible and of as high quality as possible. Sharing high-quality reusable code boosts productivity and that's what SpacePy is for. And it's all free!

Who's using it?

SpacePy is brand new, but is already being used for RBSP mission planning (see figure), and will be used in the RBSP science operations center.

At LANL there are already 12 users, and colleagues from other institutions have expressed interest.

Already one paper using SpacePy has been published and two more have been submitted.



But I use (IDL/MatLab)...

Why should I use Python?
Python is a very widely-used, modern programming language. It's free as in beer, and free as in speech. It has immense community support. It's got fast numerics, good integration for other languages (like C and FORTRAN) and is platform-independent.

How easy is it to learn?
Very. In fact, much of the plotting syntax is very similar to MatLab. One strength of Python is that it's easy to read and write – it reads almost like pseudo-code!

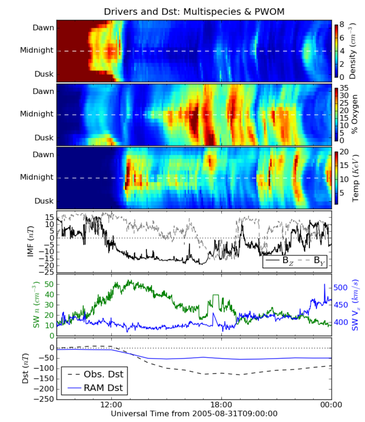
Is it interactive?
Yes. Python can run scripts on the command line, or you can work in an interactive session. A good basic shell is iPython, but IDEs like Eclipse, NetBeans and Spyder all have good Python support including tab completion, syntax highlighting, etc.

Can I read my old savefiles?
Python has good support for HDF5 (MatLab V7.3+ save files), SciPy has a function to read .mat files up to v7.1. The IDLsave module can read IDL binaries into Python. Additional formats Python supports include: netCDF, SQL, Excel, CSV, ASCII. SpacePy adds CDF support.

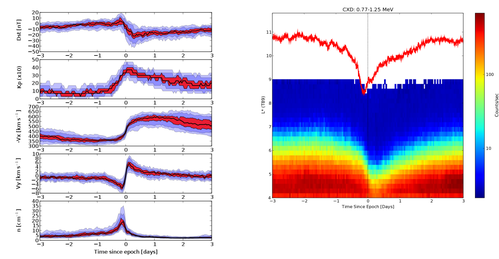
Can I make GUIs?
Yes, and it's easy! There are plenty of free tools for GUI design that will build code for Qt4, GTK+ and wxWidgets.

There's also great web and networking support and easy to use CGI and databasing support.

SpacePy in print



Inputs and boundary conditions for RAM-SCB with the polar wind outflow model. From Welling et al., submitted to JGR, 2010.

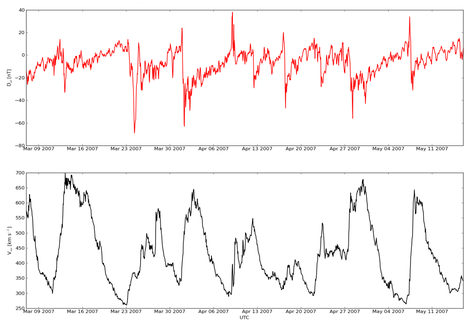


1- and 2D superposed epoch analyses of solar wind/geophysical indices for a set of 67 stream interfaces (left) and GPS-CXD electron count measurements. The 1-D plots show 95% confidence intervals for the plotted medians and IQRs. From Morley et al., Proc. Roy. Soc. A, 2010.

OMNI

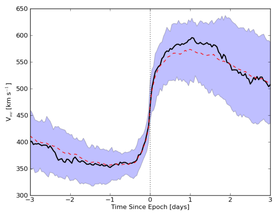
```
>>> import spacepy.omni as om
>>> import spacepy.time as spt
>>> ticks = spt.tickrange('2007-03-
07T00:00:00', '2007-05-16T00:00:00', 1./24.)
>>> data = om.get_omni(ticks)

>>> fig = figure()
>>> ax0 = fig.add_subplot(211)
>>>
ax0.plot(data['UTC'],data['Dst'],'r-',lw=1.5)
>>> ax1 = fig.add_subplot(212)
>>> ax1.plot(data['UTC'],data['velo'], 'k-',
lw=1.5)
>>> ax0.set_ylabel('D$_{st}$ [nT]')
>>> ax1.set_ylabel('V$_{sw}$ [km s$^{-1}$]')
>>> ax1.set_xlabel('UTC')
```



SeaPy

```
>>> from spacepy.seapy import *
>>> import spacepy.omni as om
>>> import spacepy.toolbox as tb
>>> epochs = readepochs('epochs.txt')
>>> st = datetime.datetime(2005,1,1)
>>> en = datetime.datetime(2009,1,1)
>>> einds, oinds = tb.tOverlap([st,
en],om.omnidata['UTC'])
>>> omni1hr = array(om.omnidata['UTC'])[oinds]
>>> delta = datetime.timedelta(hours=1)
>>> window= datetime.timedelta(days=3)
>>> sevx = se.Sea(om.omnidata['velo']
[oinds],omni1hr, epochs, window, delta)
>>> sevx.sea()
>>> sevx.plot(epochline=True, yquan='V$_{sw}$',
xunits='days', yunits='km s$^{-1}$')
```



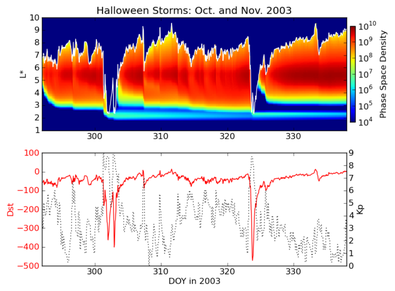
A superposed epoch analysis of the solar wind radial velocity for a set of 67 stream interfaces. The superposed epoch mean (red dashes), median (black) and IQR (blue fill) are shown.

See the "SpacePy in print" box for slightly more advanced output

RadBelt

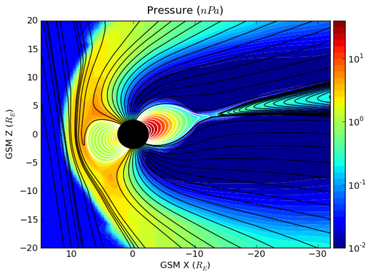
```
>>> from spacepy import radbelt as rb
>>> import datetime as dt
>>> r = rb.RBmodel()
>>> starttime = dt.datetime(2003,10,20)
>>> endtime   = dt.datetime(2003,12,5)
>>> delta = dt.timedelta(minutes=60)
>>> r.setup_ticks(starttime, endtime,
    delta, dtype='UTC')
>>> r.evolve()
>>> r.plot(clims=[4,11])
```

A 1-D radial diffusion model simulation of the Hallowe'en storms of 2003.



PyBATS

```
>>> import pybats.bats as bats
>>> obj = bats.Bats2d(`filename')
>>> obj.regrid(0.25, [-40, 15], [-30,30])
>>> obj.contourf(ax, `x', `y', `p')
>>> obj.add_body(ax)
>>> obj.add_planet_field(ax)
```



A two-dimensional slice of the simulated magnetosphere from BATS-R-US showing the plasma pressure. The black lines mark open field lines and the white lines mark closed field lines. The black circle shows the inner bounds of the simulation domain.

SpacePy has been developed and tested on both Linux (Kubuntu 9.10, 10.04 and OpenSuse 11.2) and Mac (PowerPC and Intel-based). SpacePy has been tested on a range of Python installations including the Enthought Python Distribution and Python(x,y).